Intel® Math Kernel Library (Intel® MKL) Latest Features

Sridevi Allam – Technical Consulting Engineer
Sridevi.allam@intel.com

Agenda

- Introduction to Support on Intel® Xeon Phi Coprocessors
- Performance Charts
- Link Line Advisor
- MKL 11.1 New features
- MKL 11.2 Beta New Features
- Documentation References

Intel® Math Kernel Library (Intel® MKL) Support for Intel® Xeon Phi™ Coprocessors

- Support for the Intel® Xeon Phi™ coprocessors is introduced starting Intel® MKL 11.0

- Heterogeneous computing
  -Takes advantage of both multicore host and many-core coprocessors.

- All Intel MKL functions are supported:
  -But optimized at different levels.

Highly Optimized Functions

As of Intel® Math Kernel Library 11.1 :

- BLAS Level 3, and much of Level 1 & 2

- Sparse BLAS: ?CSRMV, ?CSRMM

- Some important LAPACK routines (LU, QR, Cholesky)

- Fast Fourier transforms

- Vector Math Library

- Random number generators in the Vector Statistical Library

Usage Models on Intel® Xeon Phi™ Coprocessors

- Automatic Offload

- Compiler Assisted Offload

- Native Execution

Automatic Offload (AO)

- Offloading is automatic and transparent.
  - No code changes required
  - Automatically uses both host and target
- Can take advantage of multiple coprocessors.
- By default, Intel® Math Kernel Library decides:
  - When to offload
  - Work division between host and targets

AO Contd ..

- Users enjoy host and target parallelism automatically.

- Users can still specify work division between host and target.

Article for the List of AO Enabled Functions:
http://software.intel.com/en-us/articles/intel-mkl-automatic-offload-enabled-functions-for-intel-xeon-phi-coprocessors

How to Use Automatic Offload

- Using Automatic Offload is easy

Call a function	**Set an Env Variable**
`mkl_mic_enable()`	`MKL_MIC_ENABLE=1`

- What if there doesn’t exist a coprocessor in the system?
  - Runs on the host as usual **without penalty**!

Work Division control in AO

Examples:

mkl_mic_set_Workdivision(MKL_TARGET_MIC, 0, 0.5) : Offload 50% of computation only to the 1st Card

MKL_MIC_0_WORKDIVISION=0.5 : Offload 50% of computation only to the 1st Card

Usage Models Contd ..

Compiler Assisted Offload (CAO)

- Explicit controls of data transfer and remote execution using compiler offload pragmas/ directives
- Can be used together with Automatic Offload
- Offloading is explicitly controlled by compiler pragmas or directives.
- All Intel® Math Kernel Library (Intel® MKL) functions can be offloaded in CAO.
- Can leverage the full potential of compiler's offloading facility.

How to Use Compiler Assisted Offload

The same way you would offload any function call
to the coprocessor.
An example in C:

```
#pragma offload target(mic) \
in(transa, transb, N, alpha, beta) \
in(A:length(matrix_elements)) \
in(B:length(matrix_elements)) \
in(C:length(matrix_elements)) \
out(C:length(matrix_elements) alloc_if(0))
{
sgemm(&transa, &transb, &N, &N, &N, &alpha, A, &N, B, &N,
          &beta, C, &N);

}
```

Usage Models Contd …

Native Execution :

- Input data and binaries are copied to targets in advance

Ex: Build the code like : icc **-mmic** -mkl mkl_dft_1d.c

And manually upload the binary executable and dependent libraries to the target and ssh into target machine and run from there

- If MKL function call is inside an offload region , it consumes input and produces output only inside this offload region

Linking Examples

AO: The same way of building code on Intel® Xeon® processors!

icc -O3 -mkl sgemm.c -o sgemm.exe

Native: Using -mmic

icc -O3 **-mmic** -mkl sgemm.c -o sgemm.exe

CAO: Using -offload-option(example to Link MKL statically for both host and MIC)

icc -O3 sgemm.c -L$MKLROOT/lib/intel64 **-offload-option**, mic,ld,-L $MKLROOT/lib/mic -Wl,-Bstatic, -lmkl_intel_lp64 - Wl,--start-group -lmkl_intel_thread -lmkl_core -Wl,--end-group -Wl,-Bdynamic

Where to Find Code Examples

$MKLROOT/examples/mic_ao/blasc/source

sgemm.c -- AO Example

$MKLROOT/examples/mic_offload/.../source

sgemm.c -- blasc

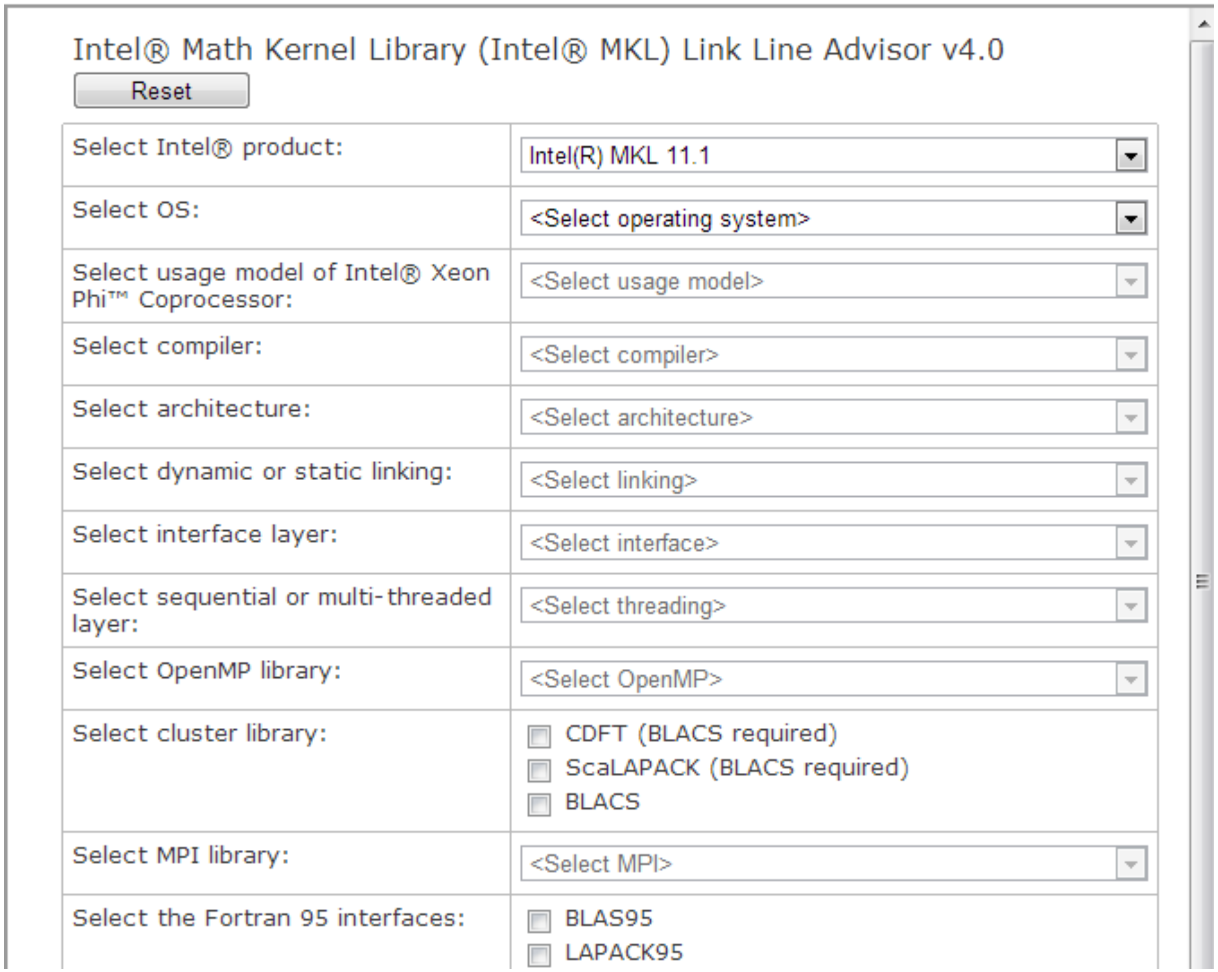
complex_dft_1d.c -- dftc

sgeqrf.c, sgetrf.c, spotrf.c -- Lapackc

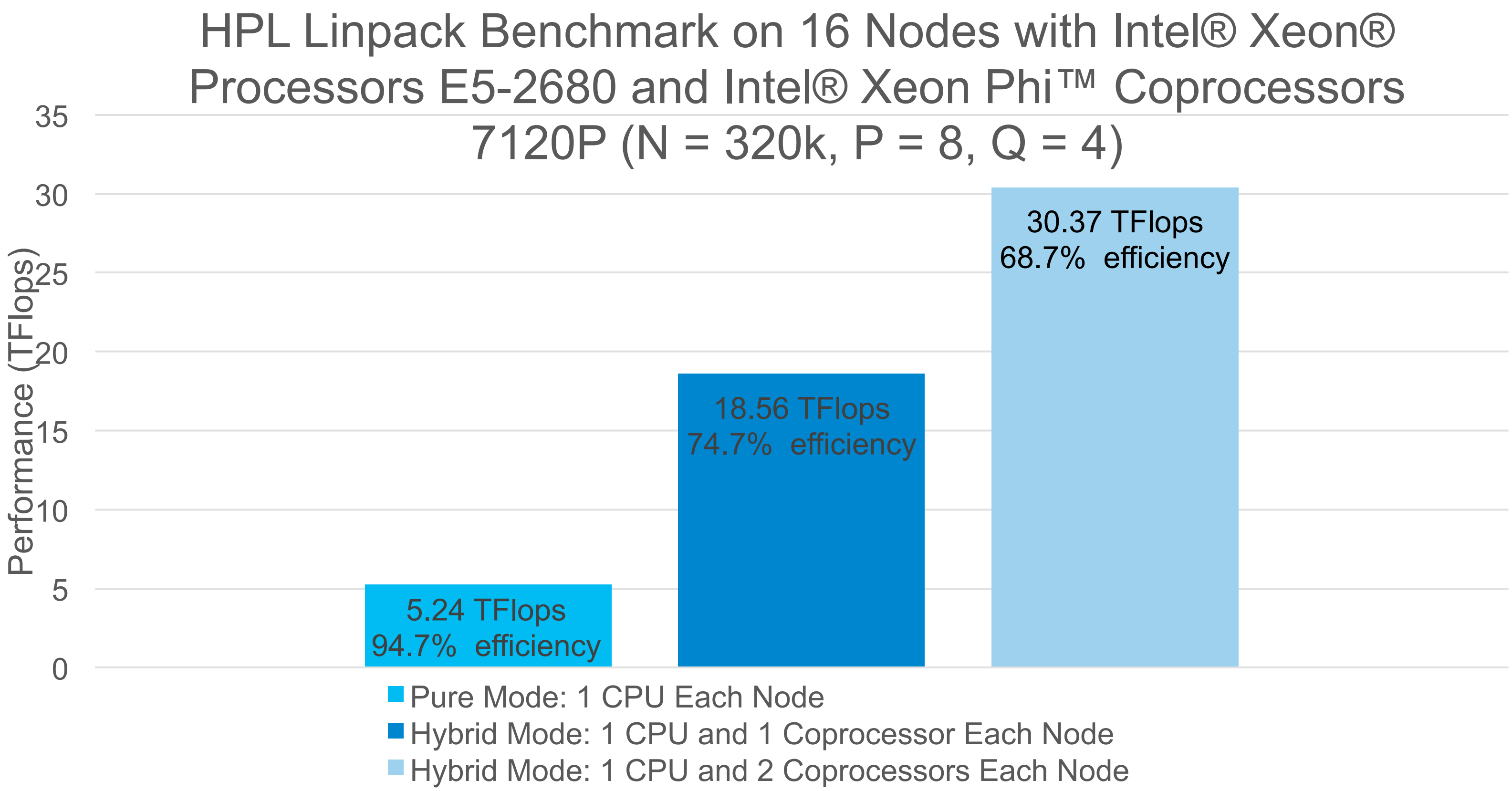
vdrnggaussian.c, vsrnggaussian.c – vslc

etc etc

Intel Math Kernel Library Link Line Advisor : http://software.intel.com/en-us/articles/intel-mkl-link-line-advisor

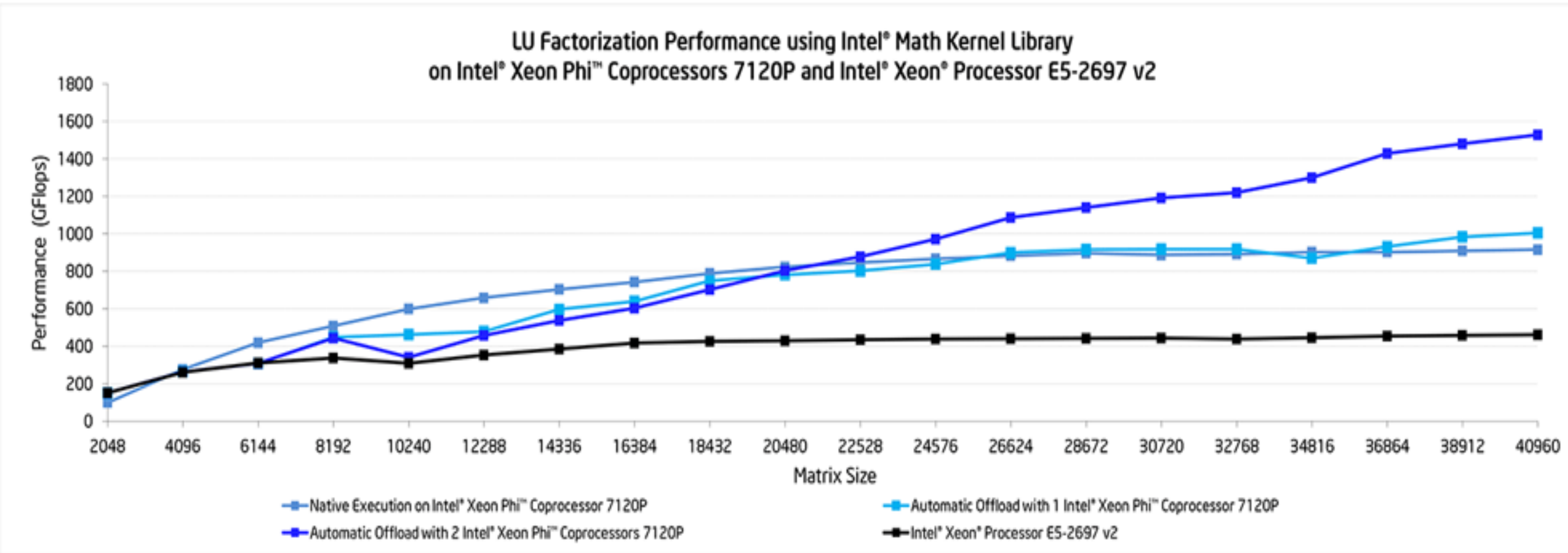


Performance Charts on Intel® Xeon Phi™ coprocessors



Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.1, Intel® MPI 4.1.0.024, Intel® C++ Compiler 13.0, Intel® Manycore Platform Software Stack (MPSS) 2.1.6720-15; Hardware of cluster nodes: Intel® Xeon® Processor E5-2680, 2 Eight-Core CPUs (20MB LLC, 2.7GHz), 64GB of RAM; Intel® Xeon Phi™ Coprocessor 7120P, 61 cores (30.5MB total cache, 1.3GHz), 16GB GDDR5 Memory; Operating System: RHEL 6.1 GA x86_64; Benchmark Source: Intel Corporation. September 2013

Performance Charts Contd …



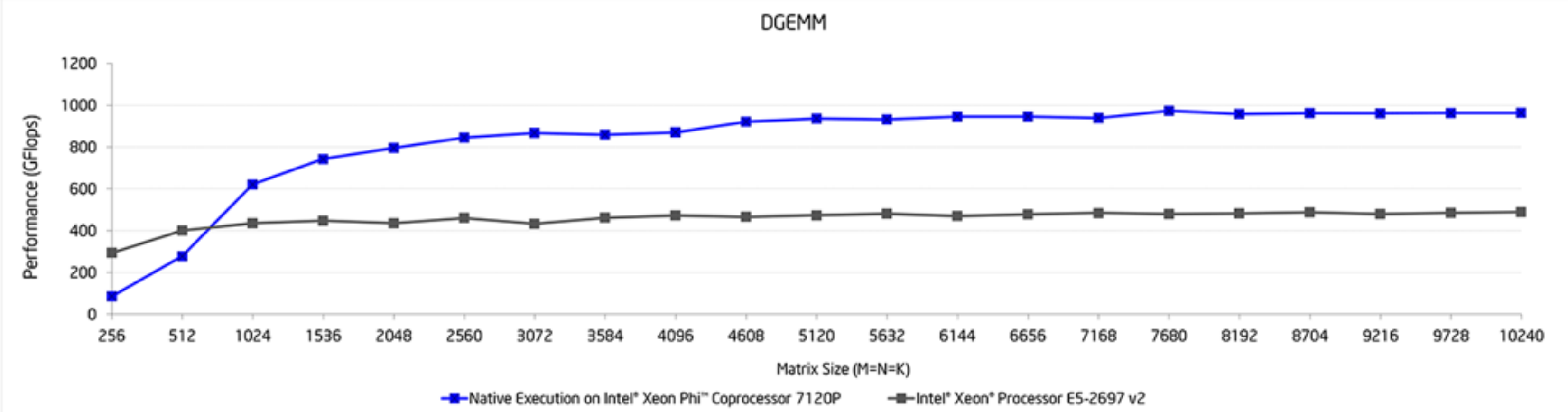
Configuration Info - Software Versions: Intel® Math Kernel Library (Intel® MKL) 11.1, Intel® Manycore Platform Software Stack (MPSS) 2.1.6720-15; Hardware: Intel® Xeon® Processor E5-2697 v2, 2 Twelve-Core CPUs (30MB LLC, 2.7GHz), 32GB DDR3 RAM (1333MHz); Intel® Xeon Phi™ Coprocessor 7120P, 61 cores (30.5MB total cache, 1.238GHz), 16GB GDDR5 Memory; Operating System: RHEL 6.1 GA x86_64.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. * Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation. September 2013.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804

Performance Charts Contd …

Matrix Multiply Performance using Intel® Math Kernel Library
on Intel® Xeon Phi™ Coprocessor 7120P and Intel® Xeon® Processor E5-2697 v2



Configuration Info - Software Versions: Intel® Math Kernel Library (Intel® MKL) 11.1, Intel® Manycore Platform Software Stack (MPSS) 2.1.6720-15; Hardware: Intel® Xeon® Processor E5-2697 v2, 2 Twelve-Core CPUs (30MB LLC, 2.7GHz), 32GB DDR3 RAM (1333MHz); vs. one Intel® Xeon Phi™ Coprocessor 7120P, 61 cores (30.5MB total cache, 1.238GHz), 16GB GDDR5 Memory; Operating System: RHEL 6.1 GA x86_64.
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. * Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation. September 2013.
Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804

Performance Tips

Problem size considerations:

-Large problems have more parallelism.

- But not too large (8GB memory on a coprocessor).
- FFT prefers power-of-2 sizes.

Data alignment consideration:

- 64-byte alignment for better vectorization.

OpenMP thread count and thread affinity:

- KMP_AFFINITY=balanced

Large (2MB) pages for memory allocation:

-Reduce TLB misses and memory allocation overhead.

http://software.intel.com/en-us/articles/performance-tips-of-using-intel-mkl-on-intel-xeon-phi-coprocessor

MKL 11.1 Highlights

- Support for Intel® Xeon Phi™ coprocessors on Windows OS* hosts.
  - The same usage models of using MKL on Linux* hosts.
- Better installation experience:
  - A choice of components to install
  - Examples and tests are packaged as archives
- HPL support for heterogeneous clusters.
- CNR support for unaligned input data.
- Performance improvements across the board.

Better installation experience

- Introduced online Installer starting with the MKL 11.1

- Introduced Partial Installation Feature:

By default, these components are NOT installed:
- Cluster components (scaLAPACK, Cluster DFT)
- Components needed by PGI* compilers (e.g. libmkl_pgi_thread.so)
- Components needed by CVF (e.g. mkl_intel_s_dll.lib)
- The SP2DP interface
- Users may re-run the installer at a later time to install any of these components.

CNR support for unaligned input data

Before

Memory alignment	• Align memory — try Intel MKL memory allocation functions • 64-byte alignment for processors in the next few years
Number of threads	• Set the number of threads to a constant number • Use sequential libraries
Deterministic task scheduling	• Ensures that FP operations occur in order to ensure reproducible results
Code path control	• Maintains consistent code paths across processors • Will often mean lower performance on the latest processors

After

Pre-requisite: Fixed number of threads	• Set the number of threads to a constant number (MKL_NUM_THREADS) • Use sequential libraries
Deterministic task scheduling	• Ensures that FP operations occur in order to ensure reproducible results
Code path control	• Maintains consistent code paths across processors • Will often mean lower performance on the latest processors

- Data alignment is no longer a requirement for getting numerical reproducibility.
- But aligning input data is still a good idea for getting better performance.

Intel® MKL 11.1 Packages

Windows*	Linux*	Mac OS* X
Intel® Parallel Studio XE Intel® C++ Studio XE Intel® Fortran Studio XE	Intel® Parallel Studio XE Intel® C++ Studio XE Intel® Fortran Studio XE	
Intel® Composer XE	Intel® Composer XE	Intel® Composer XE
Intel® C++ Composer XE	Intel® C++ Composer XE	Intel® C++ Composer XE
Intel® Fortran Composer XE Intel® Visual Fortran Composer XE	Intel® Fortran Composer XE	Intel® Fortran Composer XE
Intel® Cluster Studio XE	Intel® Cluster Studio XE	
Intel® MKL Standalone Product	Intel® MKL Standalone Product	

New features of Intel MKL 11.2 Beta

Parallel Direct Sparse Solvers for Clusters

Verbose mode for BLAS and LAPACK

S/C/Z/DGEMM improvements on small matrix sizes

Cookbook recipes

Other features and Optimizations

Parallel Direct Sparse Solvers for Clusters

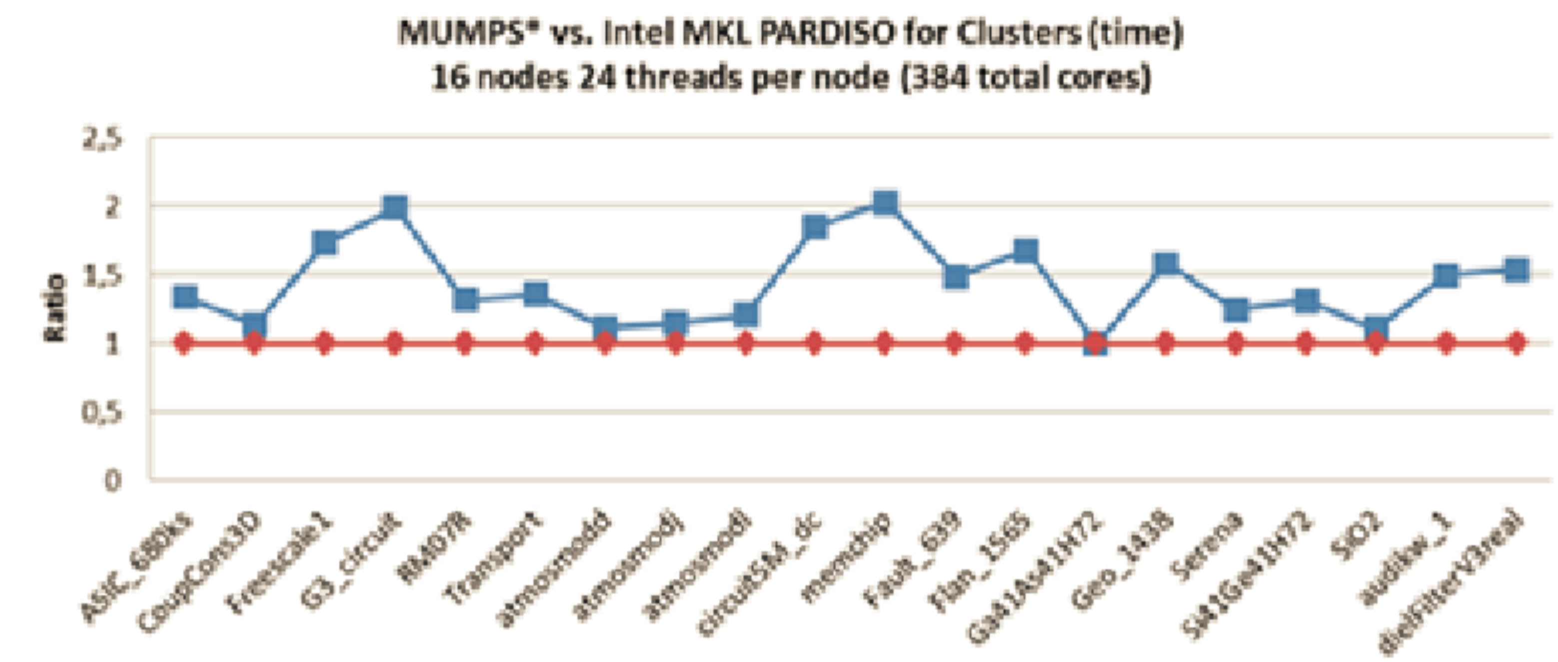
- Introduced Parallel Direct Sparse Solver for Clusters, a distributed memory version of Intel MKL PARDISO

- Parallel Direct Sparse Solver for Clusters is a Solver for a system Ax=b on a many-core cluster, where A – sparse square matrix

- It supports :
  - all types of matrixes
  - double precision only
  - LP64, LNX64, WIN64(static only)
  - any number of nodes

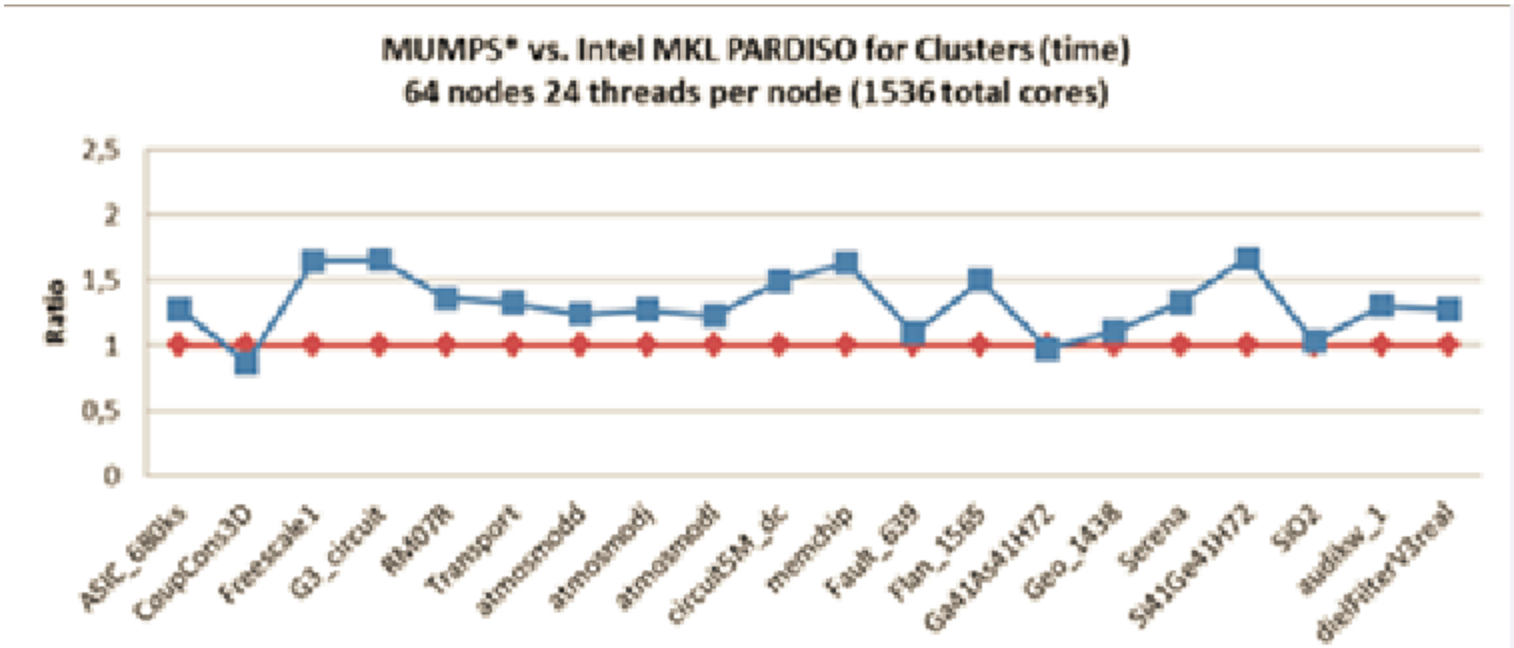
- Use both MPI & OpenMP parallelization (Hybrid parallelization)

Parallel Direct Sparse Solvers for Clusters



Numerical experiments were carried on the Infiniband*-linked cluster consisting of 96 computational nodes; each node contains two Intel®Xeon®E5-2697 v2 processors (24 cores in total) with 64Gb of RAM per node. A test was run on symmetric and unsymmetrical matrices from Florida Collection. Number of rows in matrices range from 300K to 10M. MUMPS 4.10 and MKL 11.2.b was used for testing

Parallel Direct Sparse Solvers for Clusters



Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, refer to http://www.intel.com/content/www/us/en/benchmarks/resources-benchmark-limitations.html
Refer to our Optimization Notice for more information regarding performance and optimization choices in Intel software products at: http://software.intel.com/en-ru/articles/optimization-notice/

*Other brands and names are the property of their respective owners.

Numerical experiments were carried on the Infiniband*-linked cluster consisting of 96 computational nodes; each node contains two Intel®Xeon®E5-2697 v2 processors (24 cores in total) with 64Gb of RAM per node. A test was run on symmetric and unsymmetrical matrices from Florida Collection. Number of rows in matrices range from 300K to 10M. MUMPS 4.10 and MKL 11.2.b was used for testing

Optimization Notice

Verbose Mode

How to Enable Verbose Mode? :

- Set the environment variable MKL_VERBOSE to 1
- Function Call mkl_verbose(1)

What happens when Verbose mode Enabled?

- Every call of a verbose-enabled function finishes with printing verbose log, including the list of version Information, the name of function, Values of the arguments, Time taken by the function etc

Refer to KB Article for an Example:

https://software.intel.com/en-us/articles/verbose-mode-supported-in-intel-mkl-112

Verbose Mode Example:

Running a simple cpp program for ex: icc -o dgetrf dgetrf.cpp –mkl
Here is the Output:

```
-bash-4.1$ ./dgetrf 1000
]t  all test passed
 num of host threads == 0
 ============================
floating op. counts == 666666666.666667, time_avg == 0.015514472
size == 1000, GFlops  == 42.971
```

After Setting **MKL_VERBOSE**=1, Output is as shown below:

```
-bash-4.1$ export MKL_VERBOSE=1
-bash-4.1$ ./dgetrf 1000
MKL_VERBOSE Intel(R) MKL 11.2 Beta build 20140312 for Intel(R) 64 architecture Intel(R) Advanced Vector Extens
ions (Intel(R) AVX) enabled processors, Lnx 2.70GHz lp64 intel_thread NMICDev:0
MKL_VERBOSE DLARNV(2,0x7fffd5747500,1064000,0x7f1e580f9040) 16.21ms CNR:OFF Dyn:1 FastMM:1 TID:0  NThr:16 WDiv
:HOST:+0.000MKL_VERBOSE DGETRF(1000,1000,0x7f1e580f9040,1064,0x18790f0,0) 58.55ms CNR:OFF Dyn:1 FastMM:1 TID:0
  NThr:16 WDiv:HOST:+0.000MKL_VERBOSE DGETRF(1000,1000,0x7f1e580f9040,1064,0x18790f0,0) 5.28ms CNR:OFF Dyn:1 F
astMM:1 TID:0  NThr:16 WDiv:HOST:+0.000MKL_VERBOSE DGETRF(1000,1000,0x7f1e580f9040,1064,0x18790f0,0) 5.11ms CN
R:OFF Dyn:1 FastMM:1 TID:0  NThr:16 WDiv:HOST:+0.000]t  all test passed
  num of host threads == 0
  ===========================
 floating op. counts == 666666666.666667, time_avg == 0.023002235
 size == 1000, GFlops  == 28.983
```

*GEMM Perf Improvements on Small Matrices

- Improved small S/DGEMM for Intel® AVX and AVX2
- ~1.35× improvements on average for square sizes smaller than 20
  - -Applicable to all small sizes and input parameters
- New Feature: MKL_DIRECT_CALL and MKL_DIRECT_CALL_SEQ
  - -Further improves the small matrix performance by skipping intermediate MKL function calls and error checking
  - -Additional ~1.4× improvements for S/DGEMM (square sizes smaller than 20) and ~1.15× improvements for C/ZGEMM (square sizes smaller than 8)
  - -Only static libraries and Intel Compilers are supported (certain restrictions may be removed in Intel® MKL 11.2)
  - -Compiler architecture flags (-xAVX and –xCORE-AVX2) for best performance (not necessary in Intel® MKL 11.2)
  - -See Intel® MKL User's Guide for additional details

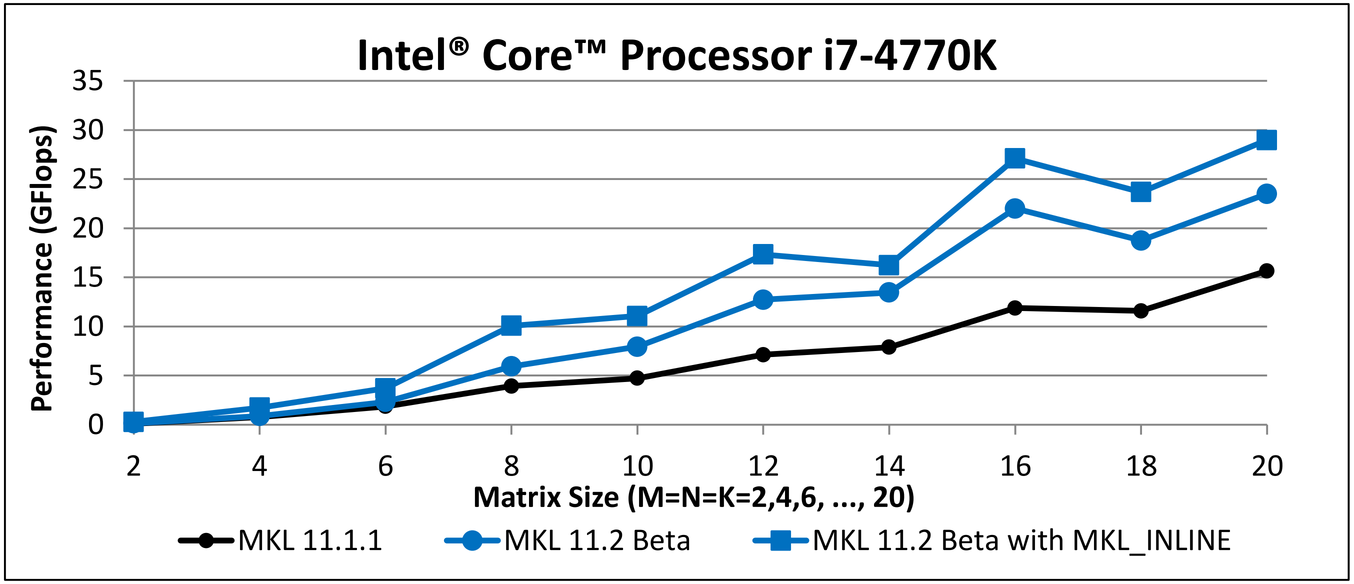
Linking Example:

- Link Line to use **MKL_DIRECT_CALL** preprocessor macro:

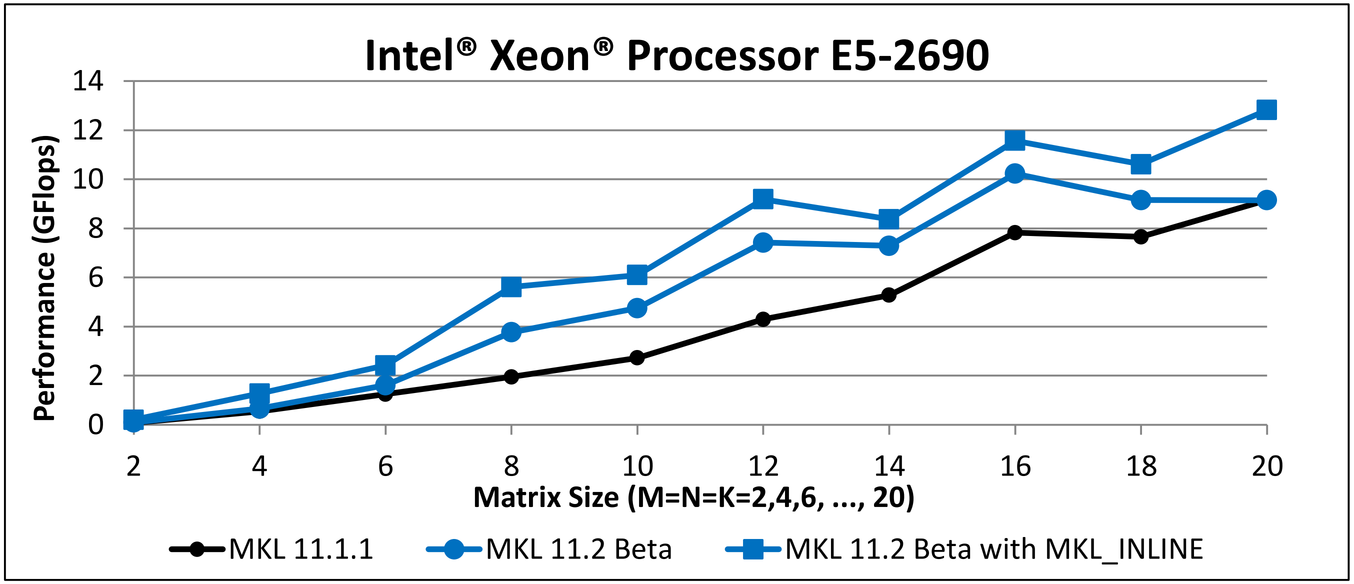
  icc –DMKL_DIRECT_CALL your_application.c -Wl,--start-group $(MKLROOT)/lib/intel64/libmkl_intel_lp64.a $(MKLROOT)/lib/intel64/libmkl_core.a $(MKLROOT)/lib/intel64/libmkl_intel_thread.a -Wl,--end-group -lpthread –lm -openmp -I$(MKLROOT)/include

- Link Line to use **MKL_DIRECT_CALL_SEQ** preprocessor macro :

  icc –DMKL_DIRECT_CALL_SEQ your_application.c -Wl,--start-group $(MKLROOT)/lib/intel64/libmkl_intel_lp64.a $(MKLROOT)/lib/intel64/libmkl_core.a $(MKLROOT)/lib/intel64/libmkl_sequential.a -Wl,--end-group -lpthread –lm -I$(MKLROOT)/include



Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.1.1 and 11.2 Beta, Intel® C++ Compiler 14.0.2; Hardware : Intel® Core™ Processor i7-4770K, Four-Core CPU (8MB LLC, 3.5 GHz), 8GB of RAM; Operating System: Fedore 19; Benchmark Source: Intel Corporation April 2014



Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.1.1 and 11.2 Beta, Intel® C++ Compiler 14.0.2; Hardware : Intel® Xeon® Processor E5-2690, 2 Eight-Core CPUs (20MB LLC, 2.9 GHz), 32GB of RAM; Operating System: RHEL 6 GA x86_64; Benchmark Source: Intel Corporation April 2014

Other features …

MKL Cookbook recipes

- Introduced Intel Math Kernel Library Cookbook, a new document with recipes for assembling Intel MKL routines for solving complex problems
http://software.intel.com/en-us/mkl_cookbook

Intel® Math Kernel Library Recipes

The Intel® Math Kernel Library (Intel® MKL) contains many routines to help you solve various numerical problems, such as multiplying matrices, solving a system of equations, and performing a Fourier transform. While many problems do not have dedicated Intel MKL routines, you can solve them by assembling the building blocks provided by Intel MKL.

The Intel Math Kernel Library Cookbook includes these recipes to help you to assemble Intel MKL routines for solving some more complex problems:

- Finding an approximate solution to a nonlinear equation demonstrates a method of finding a solution to a nonlinear equation using Intel MKL PARDISO, BLAS, and Sparse BLAS routines.
- Factoring a block tridiagonal matrix uses Intel MKL implementations of BLAS and LAPACK routines.
- Evaluating a Fourier Integral uses Fast Fourier Transforms to evaluate a continuous Fourier transform integral.
- Using Fast Fourier Transforms for computer tomography image reconstruction uses Fast Fourier Transforms to reconstruct an image from computer tomography data.
- Noise filtering in financial market data streams uses Intel MKL summary statistics routines for computing a correlation matrix for streaming data.

NOTE
Code examples in the cookbook are provided in Fortran for some recipes and in C for other recipes.

Documentation

"Using Intel® Math Kernel Library on Intel® Xeon Phi™ Coprocessors" section in the User's Guide.

http://software.intel.com/sites/products/documentation/doclib/mkl_sa/11/mkl_userguide_lnx/index.htm

"Support Functions for Intel® Many Integrated Core Architecture" section in the Reference Manual.

http://software.intel.com/en-us/node/468334#D6B418C3-90EA-4431-94DB-124780171AD6

Intel® Compiler 13.0 User Guide and Reference Manual.

http://software.intel.com/en-us/node/458836#2632E0AD-C8CF-427C-802B-52A06AC778F2

Intel MKL 11.2 Beta References

Intel MKL 11.2 Beta release notes:

https://software.intel.com/en-us/articles/intel-mkl-112-release-notes

Intel MKL 11.2 Beta Reference Manual and Articles:
https://software.intel.com/en-us/articles/intel-mkl-112-beta

Intel MKL Product Page:
https://software.intel.com/en-us/intel-mkl

Online Resources

Articles, tips, case studies, hands-on lab:
http://software.intel.com/en-us/articles/intel-mkl-on-the-intel-xeon-phi-coprocessors

Performance charts online:
http://software.intel.com/en-us/intel-mkl#pid-12780-836

The MIC developer community:
http://www.intel.com/software/mic-developer

Intel® Math Kernel Library forum: http://software.intel.com/en-us/forums/intel-math-kernel-library

Legal Disclaimer & Optimization Notice

Optimization Notice

Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804